Assignment-4(SMSSPAMClassification)

1. StudentName: Sowmya J
2. RegisterNumber:711519BEC309
3. TeamID:PNT2022TMID07418

import numpy as npimportpandasaspd

importseabornas sns

importmatplotlib.pyplotas plt

data = pd.read\_csv('/content/sample\_data/spam.csv',delimiter=',',encoding='latin-1')data.head()

**v1 v2 Unnamed:**

**2**

**Unnamed:**

**3**

**Unnamed:**

**4**

* 1. ham Gountiljurongpoint,crazy..Availableonly... NaN NaN NaN
  2. ham Oklar...Jokingwifuoni... NaN NaN NaN
  3. spam Freeentryin2awklycomptowinFACup

fina...

* 1. ham Udunsaysoearlyhor...Ucalreadythen

say...

NaN NaN NaN

NaN NaN NaN

data.columns

Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')

#drop the unamed columns

data=data.drop(columns=["Unnamed:2","Unnamed:3","Unnamed:4"])

#rename the two relevant columnsdata=data.rename(

{

"v1":"Category",

"v2":"Message"

},axis=1)

data.head()

**Category Message **

1. ham Gountiljurongpoint,crazy..Availableonly...
2. ham Oklar...Jokingwifuoni...
3. spam Freeentryin2awklycomptowinFACupfina...

#check for null valuesdata.isnull().sum()

**3** ham Udunsaysoearlyhor...Ucalreadythensay...

**4** ham NahIdon'tthinkhegoestousf,helivesaro...

Category 0

Message 0

dtype: int64

data.info()

<class 'pandas.core.frame.DataFrame'>RangeIndex: 5572 entries, 0 to 5571

Data columns (total 2 columns):

# Column Non-Null CountDtype

1. Category5572 non-null object
2. Message 5572 non-null objectdtypes: object(2)

memory usage: 87.2+ KB

data["MessageLength"]=data["Message"].apply(len)

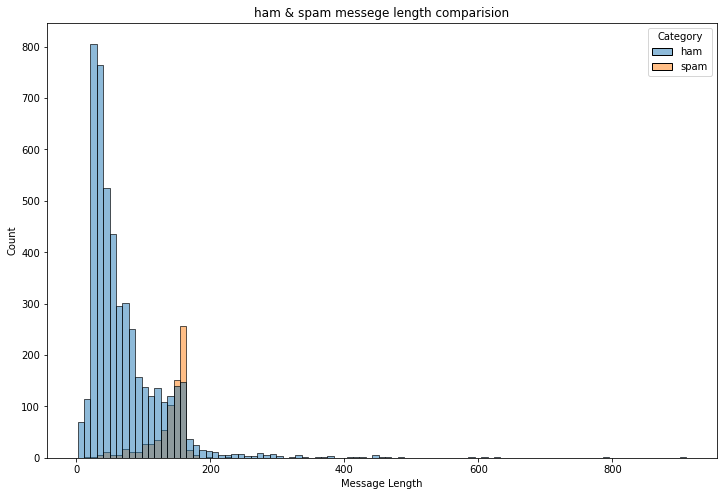
fig=plt.figure(figsize=(12,8))sns.histplot(

x=data["Message Length"],hue=data["Category"]

)

plt.title("ham & spam messege length comparision")plt.show()

#Display the description of length of ham and spam messages seperately on an individual se



ham\_desc=data[data["Category"]=="ham"]["Message Length"].describe()spam\_desc=data[data["Category"]=="spam"]["Message Length"].describe()print("HamMessege Length Description:\n",ham\_desc)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Spam Message Length Description:\n",spam\_desc)Ham Messege Length Description:

|  |  |
| --- | --- |
| count | 4825.000000 |
| mean | 71.023627 |
| std | 58.016023 |
| min | 2.000000 |
| 25% | 33.000000 |
| 50% | 52.000000 |
| 75% | 92.000000 |
| max | 910.000000 |
| Name: | Message Length, dtype: float64 |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spam Message Length Description:

|  |  |
| --- | --- |
| count | 747.000000 |
| mean | 138.866131 |
| std | 29.183082 |
| min | 13.000000 |
| 25% | 132.500000 |
| 50% | 149.000000 |
| 75% | 157.000000 |
| max | 224.000000 |
| Name: | Message Length, dtype: float64 |

data.describe(include="all")



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Category** | **Message** | **Message Length** |
| **count** | 5572 | 5572 | 5572.000000 |
| **unique** | 2 | 5169 | NaN |

data["Category"].value\_counts()

**top** ham Sorry,I'llcalllater NaN

**freq** 4825 30 NaN

**mean** NaN NaN 80.118808

**std** NaN NaN 59.690841

ham 4825

spam 747

Name: Category, dtype: int64

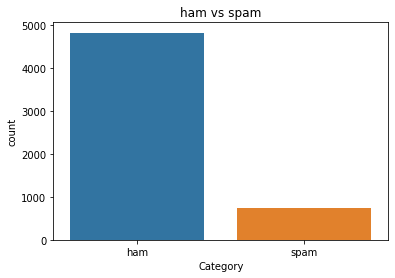
**min** NaN NaN 2.000000

**25%** NaN NaN 36.000000

**50%** NaN NaN 61.000000

**75%** NaN NaN 121.000000

**max** NaN NaN 910.000000



sns.countplot(data=data,x="Category"

)

plt.title("ham vs spam")plt.show()

ham\_count=data["Category"].value\_counts()[0]spam\_count=data["Category"].value\_counts()[1]

total\_count=data.shape[0]

print("Ham contains:{:.2f}% of total data.".format(ham\_count/total\_count\*100))print("Spamcontains:{:.2f}%oftotaldata.".format(spam\_count/total\_count\*100))

Ham contains:86.59% of total data.Spamcontains:13.41%oftotaldata.

#compute the length of majority & minority classminority\_len=len(data[data["Category"]=="spam"])majority\_len=len(data[data["Category"]=="ham"])

#store the indices of majority and minority class

minority\_indices=data[data["Category"]=="spam"].indexmajority\_indices=data[data["Category"]=="ham"].index

#generatenew majority indices from the total majority\_indices

#with size equal to minority class length so we obtain equivalent number of indices lengthrandom\_majority\_indices=np.random.choice(

majority\_indices,size=minority\_len,replace=False

)

#concatenate the two indices to obtain indices of new dataframe

undersampled\_indices=np.concatenate([minority\_indices,random\_majority\_indices])

#create df using new indices

df=data.loc[undersampled\_indices]

#shuffle the sampledf=df.sample(frac=1)

#reset the index as its all mixeddf=df.reset\_index()

#drop the older indexdf=df.drop(

columns=["index"],

)

df.shape

(1494, 3)

df["Category"].value\_counts()ham 747

spam 747

Name: Category, dtype: int64

sns.countplot(data=df,

x="Category"

)

plt.title("ham vs spam")plt.show()

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Message** | **Message** | **Length** |
| **0** ham | Aah! A cuddle would be lush! I'd need lots of ... |  | 87 |
| **1** ham | I'm in solihull, | do you want anything? |  | 40 |
| **2** spam | DoubleMins &1000 txtson Orangetariffs. Lat... |  | 151 |
| **3** ham | No we put party 7 days a week and study lightl... |  | 126 |
| **4** spam | URGENT!!Your4\*CostaDelSolHolidayorå£50... |  | 161 |

#Created new column Label and encode ham as 0 and spam as 1df["Label"]=df["Category"].map(



df.head()

{

"ham":0,

"spam":1

}

)

df.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Message** | **Message** | **Length** | **Label** |
| **0** ham | Aah! A cuddle would be lush! I'd need lots of ... |  | 87 | 0 |
| **1** ham | I'm in solihull, | do you want anything? |  | 40 | 0 |
| **2** spam | DoubleMins &1000 txtson Orangetariffs. Lat... |  | 151 | 1 |
| **3** ham | No we put party 7 days a week and study lightl... |  | 126 | 0 |
| **4** spam | URGENT!!Your4\*CostaDelSolHolidayorå£50... |  | 161 | 1 |

importre

importnltk

from nltk.corpus import stopwordsfromnltk.stemimportPorterStemmer

stemmer=PorterStemmer()

nltk.download('stopwords')

[nltk\_data] Downloading package stopwords to /root/nltk\_data...[nltk\_data] Unzipping corpora/stopwords.zip.

True

#declare empty list to store tokenized messagecorpus=[]

#iterate through the df["Message"]formessageindf["Message"]:

#replace every special characters, numbers etc.. with whitespace of message#It will help retain only letter/alphabets

message=re.sub("[^a-zA-Z]","",message)

#convert every letters to its lowercasemessage=message.lower()

#split the word into individual word listmessage=message.split()

#perform stemming using PorterStemmer for all non-english-stopwordsmessage=[stemmer.stem(words)

forwordsin message

ifwordsnotinset(stopwords.words("english"))

]

#join the word lists with the whitespacemessage="".join(message)

#append the message in corpus listcorpus.append(message)

from tensorflow.keras.preprocessing.text import one\_hotvocab\_size=10000

oneHot\_doc=[one\_hot(words,n=vocab\_size)forwordsin corpus

]

df["MessageLength"].describe()

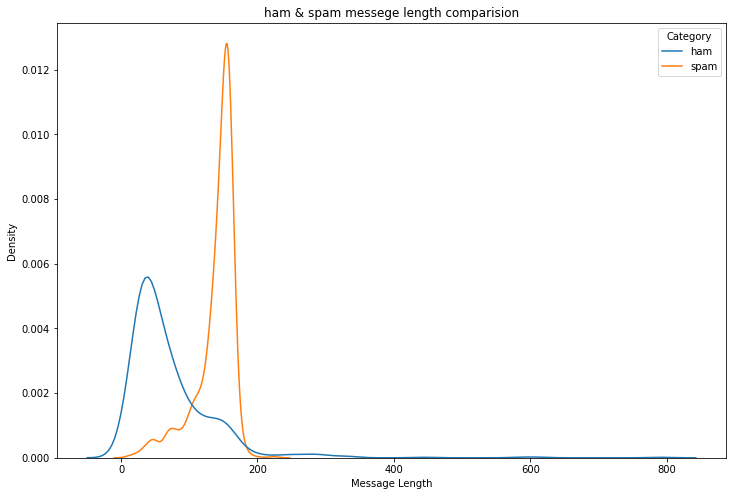
|  |  |  |
| --- | --- | --- |
| count | 1494.000000 |  |
| mean | 105.203481 |  |
| std | 61.166448 |  |
| min | 3.000000 |  |
| 25% | 48.000000 |  |
| 50% | 118.000000 |  |
| 75% | 153.000000 |  |
| max  Name: | 790.000000  Message Length, dtype: | float64 |

fig=plt.figure(figsize=(12,8))sns.kdeplot(

x=df["Message Length"],hue=df["Category"]

)

plt.title("ham & spam messege length comparision")plt.show()



from tensorflow.keras.preprocessing.sequence import pad\_sequencessentence\_len=200

embedded\_doc=pad\_sequences(oneHot\_doc,

maxlen=sentence\_len,padding="pre"

)

extract\_features=pd.DataFrame(data=embedded\_doc

)

target=df["Label"]

df\_final=pd.concat([extract\_features,target],axis=1)

df\_final.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9... 191** | **192** | **193** | **194** | **195** | **196** | **197** | **198** |
| **0** 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ... 2090 | 1632 | 4289 | 7158 | 478 | 5808 | 6133 | 8348 |
| **1** 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ... 0 | 0 | 0 | 0 | 0 | 0 | 8663 | 4425 |

X=df\_final.drop("Label",axis=1)y=df\_final["Label"]

**2** 0 0 0 0 0 0 0 0 0 0 ... 1275 702 1694 4114 4162 3935 4162 8536

**3** 0 0 0 0 0 0 0 0 0 0 ... 3705 9946 5462 7158 9883 4500 8030 8630

**4** 0 0 0 0 0 0 0 0 0 0 ... 4753 6414 5018 1953 216 1175 8861 2485

5rows×201columns

fromsklearn.model\_selectionimporttrain\_test\_split

X\_trainval,X\_test,y\_trainval,y\_test=train\_test\_split(X,

y,

random\_state=42,test\_size=0.15

)

X\_train,X\_val,y\_train,y\_val=train\_test\_split(X\_trainval,

y\_trainval,

random\_state=42,test\_size=0.15

)

from tensorflow.keras.layers import LSTMfromtensorflow.keras.layersimportDense

from tensorflow.keras.layers import Embeddingfrom tensorflow.keras.models import Sequentialmodel=Sequential()

feature\_num=100model.add(

Embedding(

input\_dim=vocab\_size,output\_dim=feature\_num,

input\_length=sentence\_len

)

)

model.add(

LSTM(

units=128

)

)

model.add(

Dense(

units=1,

activation="sigmoid"

)

)

from tensorflow.keras.optimizers import Adammodel.compile(

optimizer=Adam(

learning\_rate=0.001

),

loss="binary\_crossentropy",metrics=["accuracy"]

)

model.fit(

X\_train,y\_train,

validation\_data=(X\_val,

y\_val

),

epochs=10

)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Epoch | 1/10 |  | | | | | |
| 34/34  Epoch34/34 | [==============================]2/10  [==============================] | * 8s * 1s | 33ms/step  16ms/step | * loss: * loss: | 0.5258  0.1718 | * accuracy: * accuracy: | 0.7  0.9 |
| Epoch  34/34 | 3/10  [==============================] | - 1s | 16ms/step | - loss: | 0.0533 | - accuracy: | 0.9 |
| Epoch | 4/10 |  |  |  |  |  |  |
| 34/34  Epoch34/34 | [==============================]5/10  [==============================] | * 1s * 1s | 15ms/step  16ms/step | * loss: * loss: | 0.0254  0.0184 | * accuracy: * accuracy: | 0.9  0.9 |
| Epoch34/34 | 6/10  [==============================] | - 1s | 16ms/step | - loss: | 0.0134 | - accuracy: | 0.9 |
| Epoch  34/34 | 7/10  [==============================] | - 1s | 16ms/step | - loss: | 0.0150 | - accuracy: | 0.9 |
| Epoch | 8/10 |  |  |  |  |  |  |
| 34/34  Epoch34/34 | [==============================]9/10  [==============================] | * 1s * 1s | 16ms/step  16ms/step | * loss: * loss: | 0.0112  0.0062 | * accuracy: * accuracy: | 0.9  0.9 |
| Epoch  34/34 | 10/10  [==============================] | - 1s | 16ms/step | - loss: | 0.0050 | - accuracy: | 0.9 |

<keras.callbacks.History at 0x7fa3263a7850>

y\_pred=model.predict(X\_test)y\_pred=(y\_pred>0.5)

8/8 [==============================] - 0s 8ms/step

fromsklearn.metricsimportaccuracy\_score,confusion\_matrix

score=accuracy\_score(y\_test,y\_pred)

print("TestScore:{:.2f}%".format(score\*100))

Test Score:96.00%

cm=confusion\_matrix(y\_test,y\_pred)fig=plt.figure(figsize=(12,8))

sns.heatmap(cm,

annot=True,

)

plt.title("Confusion Matrix")cm

array([[100, 2],

[7, 116]])



#The function take model and message as parameterdefclassify\_message(model,message):

#We will treat message as a paragraphs containing multiple sentences(lines)#we will extract individual lines

forsentencesin message:

sentences=nltk.sent\_tokenize(message)

#Iterate over individual sentencesforsentence in sentences:

#replace all special characters

words=re.sub("[^a-zA-Z]","",sentence)

#perform word tokenization of all non-english-stopwordsifwordsnotinset(stopwords.words('english')):

word=nltk.word\_tokenize(words)word="".join(word)

#perform one\_hot on tokenized wordoneHot=[one\_hot(word,n=vocab\_size)]

#create an embedded documnet using pad\_sequences#this can be fed to our model

text=pad\_sequences(oneHot,maxlen=sentence\_len,padding="pre")

#predict the text using modelpredict=model.predict(text)

#if predict value is greater than 0.5 its a spamifpredict>0.5:

print("Itisaspam")

#else the message is not a spamelse:

print("Itisnot aspam")

message1="Iam having a bad day and I would like to have a break today"

message2="Thisis to inform you had won a lotteryand the subscription will end in a week

nltk.download('punkt')

[nltk\_data] Downloading package punkt to /root/nltk\_data...[nltk\_data] Unzipping tokenizers/punkt.zip.

True

classify\_message(model,message1)

1/1 [==============================] - 0s 21ms/step

It is not a spam

classify\_message(model,message2)

1/1 [==============================] - 0s 22ms/step

It is a spam

[Colabpaidproducts](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links)-[Cancelcontractshere](https://colab.research.google.com/cancel-subscription)0s completedat8:07AM